vicomtech

your R&D partner
for smart digital solutions

Intelligent Transport Systems & Engineering

Dr. Marcos Nieto
mnieto@vicomtech.org

vicomtech

OpenLABEL – Data Format

- The **annotation format** defines the structure of annotations, data types and conventions needed to unambiguously interpret the annotations.

- The **annotation file format** specifies how the annotation data is encoded for storage into computer files.

- A **JSON schema** is used to define the structure of JSON (JavaScript Object Notation) data for validation.

https://code.asam.net/simulation/standard/openlabel-schema/-/blob/master/openlabel_json_schema-v0.2.0.json

vicomtech

JSON SCHEMA

OpenLABEL – Data format

JSON Schema

The Schema **defines**:

- -The terms of labels (**keys**)
- -The types of the **values**
- -The structure/hierarchy
- -Restrictions (required/optional) and conditions

The Schema **is used to**:

- -Validate JSON files
- -Provide documentation
- -Be parsed into programming languages to build classes

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "additionalProperties": false,
    "definitions": {
        "action": {
            "additionalProperties": false,
            "properties": {
                "frame_intervals": {
                    "item": {
                        "$ref": "#/definitions/frame_interval"
                    },
                    "type": "array"
                },
                "name": {
                    "type": "string"
                },
                "ontology_uid": {
                    "type": "integer"
                },
                "stream": {
                    "type": "string"
                },
                "type": {
                    "type": "string"
                }
            },
            "required": [
                "name",
                "type"
            ],
            "type": "object"
        },
        "area_reference": {
            "properties": {
                "additionalProperties": false,
                "attributes": {
                    "$ref": "#/definitions/attributes"

...
```

JSON schema snippet

vicomtech

JSON STRUCTURE

OpenLABEL – Data format

OpenLabel JSON

```
{
    "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        }
    }
}
```

"openlabel" root key

OpenLABEL – Data format

OpenLabel root level

```
{
    "openlabel": {
        "objects": { ... },
        "actions": { ... },
        "events": { ... },
        "contexts": { ... },
        "relations": { ... },
        "frames": { ... },
        "frame_intervals": { ... },
        "tags": { ... },
        "metadata": { ... },
        "ontologies": { ... },
        "resources": { ... },
        "coordinate_systems": { ... },
        "streams": { ... }
    }
}
```

Keys defined in **JSON schema**

Values subject to schema types and properties

OpenLABEL – Data format

OpenLabel JSON simple **object**

```
{
    "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "objects": {
            "0": {
                "name": "car1",
                "type": "Car",
            }
        }
    }
}
```

uid unique identifier as numerical string

name of the instance
type of the class

```
{
   "openlabel": {
       "metadata": {
           "schema_version": "0.2.0"
       },
       "objects": {
          "c44c1fc2-ee48-4b17-a20e-829de9be1141": {
              "name": "van1",
              "type": "Van"
          },
       }
   }
}
```

uuid Universal Unique Identifier

OpenLABEL – Data format

OpenLabel JSON simple **object** with **object_data**

```
{
   "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "objects": {
           "0": {
               "name": "pedestrian1",
               "type": "Pedestrian",
               "object_data": {
                   "bbox" : [{
                               "name" : "body",
                               "val" : [303.73, 935.58, 135.62, 330.88]
                           }, {
                               "name" : "head",
                               "val" : [289.93, 814.08, 38.20, 39.96]
                           }
                    ]
               }
           }
        }
   }
}
```

bbox array, to allow multiple entries per object

object_data contains data about object

name of bbox
val (x, y, w, h)

OpenLABEL – Data format

OpenLabel JSON simple **object** with **object_data** with **attributes**

```
{
    "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "objects": {
            "0": {
                "name": "car1",
                "type": "Car",
                "object_data": {
                    "bbox" : [{
                                "name" : "shape",
                                "val" : [100, 100, 500, 300],
                                "attributes": {
                                    "boolean": [{
                                                "name": "visible",
                                                "val": true
                                            },
                                            {
                                                "name": "interpolated",
                                                "val": false
                                            }
                                        ]
                                }
                            }
                        ]
                }
            }
        }
    }
}
```

object_data contains data about object

attributes of bbox

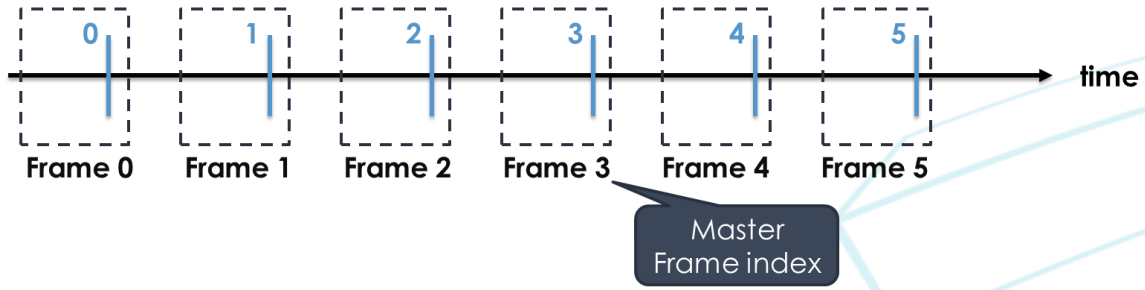
OpenLABEL – Data format

OpenLabel JSON simple **frames**

```
{
   "openlabel": {
       "metadata": {
           "schema_version": "0.2.0"
       },
       "frame_intervals": [{
               "frame_start": 0, "frame_end": 1
           }, {
               "frame_start": 5, "frame_end": 7
           }
       ],
       "frames": {
           "0": { ... },
           "1": { ... },
           "5": { ... },
           "6": { ... },
           "7": { ... }
       }
   }
}
```

frames contains dictionary of frames

Frame_intervals ranges of frame numbers with data



OpenLABEL – Data format

OpenLabel JSON empty **frames**

```
{
   "openlabel": {
       "metadata": {
            "schema_version": "0.2.0"
        },
       "frames": {
           "0": {
               "objects": {
                   "1": {}
               }
           },
           "1": {
               "objects": {
                   "1": {}
               }
           }
       },
       "objects": {
           "1": {
               "name": "van1",
               "type": "Van",
               "frame_intervals": [{"frame_start": 0, "frame_end": 1}]
           },
       }
   }
}
```

frame contains dynamic info of objects

{ } void entry means, "object exists at this frame, but no dynamic data specified"

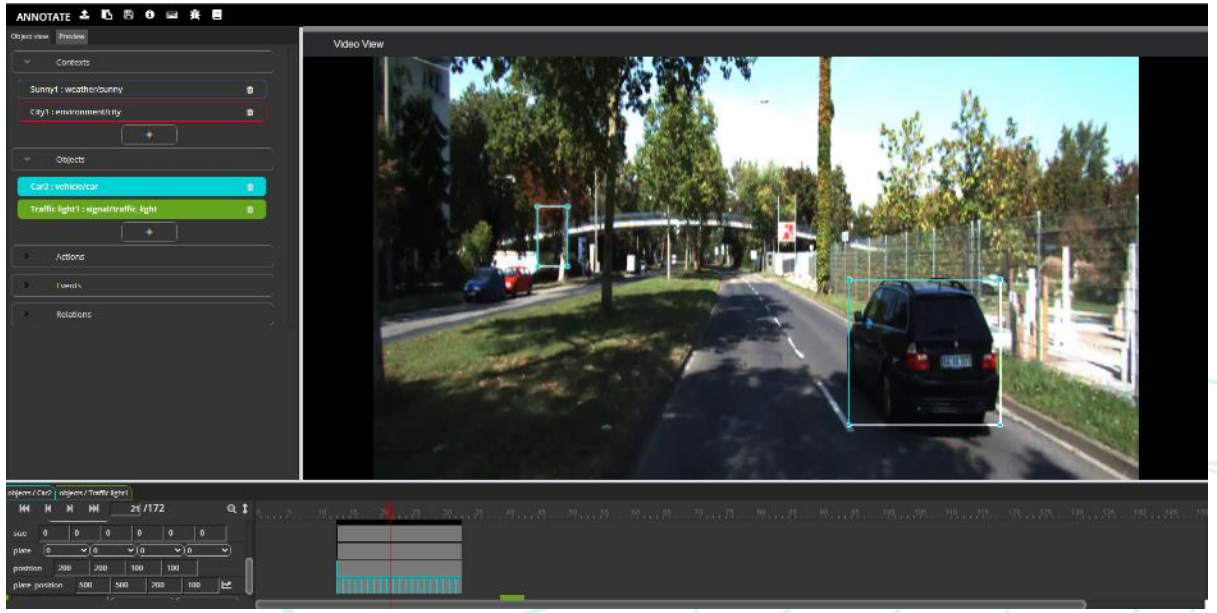
objects contains static information of objects

OpenLABEL – Data format

OpenLabel JSON **object_data** at **frame**

```
{
    "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "frames": {
            "0": {
                "objects": {
                    "1": {
                        "object_data": {
                            "bbox": [{
                                    "name": "shape",
                                    "val": [12, 867, 600, 460]
                                }
                            ]
                        }
                    }
                }
            },
            "1": { ... }
        },
        "objects": {
            "1": {
                "name": "van1",
                "type": "Van",
                "frame_intervals": [{"frame_start": 0, "frame_end": 1}]
            },
        }
    }
}
```

bbox specified for this frame ("0"), for this object ("1")



OpenLABEL – Data format

OpenLabel JSON object_data_pointers

```
{
   "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "objects": {
           "0": {
               "name": "car0",
               "type": "car",
               "frame_intervals": [{"frame_start": 0, "frame_end": 10}],
               "object_data": {
                   "text": [{
                           "name": "color",
                           "val": "blue"
                       }
                   ],
               },
               "object_data_pointers": {
                   "color": {
                       "type": "text",
                   },
                   "shape": {
                       "type": "bbox",
                       "frame_intervals": [{"frame_start": 0, "frame_end": 10}],
                       "attributes": {
                           "visible": "boolean"
                       }
                   }
               }
           }
        },
        "frames": {
            "0": { ... },
            ...
            "10": { ... }
        }
        ...
    }
}
```

text specified static

Pointer to static object_data

Pointer to dynamic object_data

object_data_pointers as summary of static and dynamic object_data, indexed by "name"

object contains dynamic data at frames

OpenLABEL – Data format

OpenLabel JSON **streams**

```
{
   "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "streams": {
            "Camera1": {
                "type": "camera",
                "uri": "./some_path/some_video.mp4",
                "description": "Frontal camera",
                "stream_properties": {
                    "intrinsics_pinhole": {
                        "camera_matrix_3x4": [ 1000.0,    0.0, 500.0, 0.0,
                                               0.0, 1000.0, 500.0, 0.0,
                                               0.0,    0.0,   0.0, 1.0],
                        "distortion_coeffs_1xN": [],
                        "height_px": 480,
                        "width_px": 640
                    },
                }
            }
        }
    }
}
```

streams contain dictionary of sequences of data subject of labeling

stream_properties define properties of stream **"Camera1"**

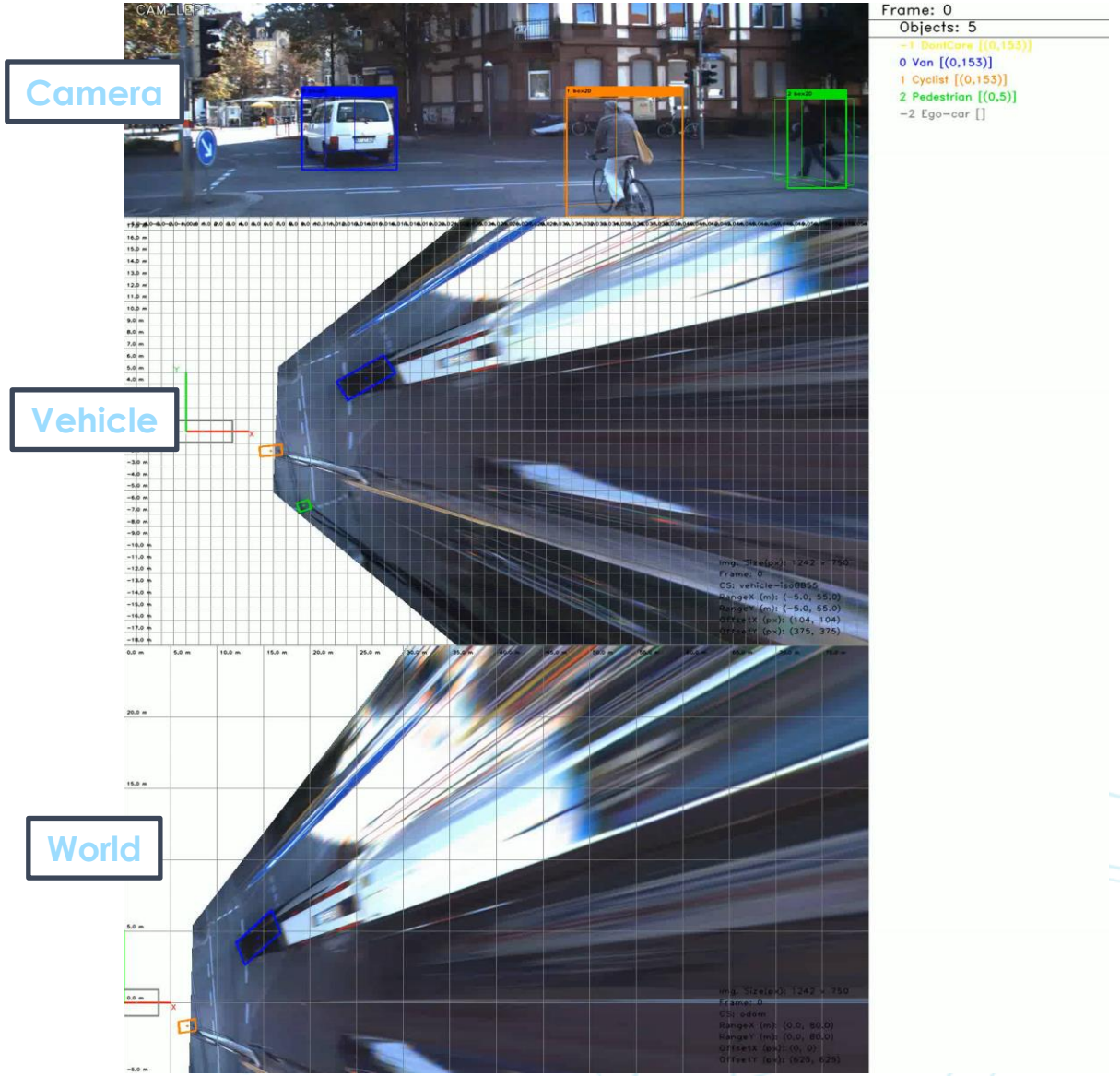
OpenLABEL – Data format

OpenLabel JSON coordinate_systems

```
{
    "openlabel": {
        "metadata": {
            "schema_version": "0.2.0"
        },
        "coordinate_systems": {
            "odom": {
                "type": "scene_cs",
                "parent": "",
                "children": [
                    "vehicle-iso8855"
                ]
            },
            "vehicle-iso8855": {
                "type": "local_cs",
                "parent": "odom",
                "children": [
                    "CAM_1",
                    "CAM_2"
                ]
            },
            "CAM_1" : {
                "type" : "sensor_cs",
                "parent" : "base",
                "children" : [],
                "pose_wrt_parent" : {
                    "matrix4x4" : [0.984807753012208, 0.0, 0.17364817766693033, 2.3, 0.0, 1.0, 0.0, 0.0, -
0.17364817766693033, 0.0, 0.984807753012208, 1.3, 0.0, 0.0, 0.0, 1.0]
                }
            },
            "CAM_2" : {
                "type" : "sensor_cs",
                "parent" : "base",
                "children" : [],
                "pose_wrt_parent" : {
                    "euler_angles" : [0.0, 0.17453292519943295, 0.0],
                    "translation" : [2.3, 0.0, 1.5],
                    "sequence" : "ZYX"
                }
            }
        },
        ...
    }
}
```

coordinate_systems contain dictionary of defined coordinate systems

Hierarchy of "parent" and "children" poses



OpenLABEL – Data format

OpenLabel JSON transforms

```
{
    "openlabel" : {
        "coordinate_systems" : {
            "base" : {
                "type" : "local_cs",
                "parent" : "",
                "children" : []
            },
            "world" : {
                "type" : "scene_cs",
                "parent" : "",
                "children" : []
            }
        },
        "frames" : {
            "10" : {
                "frame_properties" : {
                    "transforms" : {
                        "base_to_world" : {
                            "src" : "base",
                            "dst" : "world",
                            "transform_src_to_dst" : {
                                "matrix4x4" : [1.0, 0.0, 0.0, 0.1, 0.0, 1.0, 0.0, 0.1, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0]
                            }
                        }
                    }
                }
            },
            "11" : {
                "frame_properties" : {
                    "transforms" : {
                        "base_to_world" : {
                            "src" : "base",
                            "dst" : "world",
                            "transform_src_to_dst" : {
                                "euler_angles" : [0.0, 0.0, 0.0],
                                "translation" : [1.0, 1.0, 0.0],
                                "sequence" : "ZYX"
                            },
                            "custom_property1" : 0.9,
                            "custom_property2" : "Some tag"
                        }
                    }
                }
            }
        }
    }
}
```

transforms define frame-specific conversion from one coordinate system to another

Several supported format of transforms: **"matrix4x4"**, **"euler_angles"**, **"quaternion"**

vicomtech

DATA TYPES

OpenLABEL – Data format

OpenLabel JSON generic data types: **object_data**

```
"num": [{
    "name": "height_m",
    "val": 1.98
}]
```

```
"text": [{
    "name": "license plate",
    "val": "8440CMN"
}]
```

```
"vec": [{
    "name": "scores",
    "val": [0.98, 0.76, 0.98]
}]
```

```
"vec": [{
    "name": "locations",
    "val": ["Madrid", "Paris", "Rome"]
}]
```

```
"boolean": [{
    "name": "visible",
    "val": true
}]
```

OpenLABEL – Data format

OpenLabel JSON geometric data types: object_data

```
"bbox": [{
    "name": "head",
    "val": [400, 200, 100, 120]
}]
```

```
"rbbox": [{
    "name": "outline",
    "val": [400, 200, 100, 120, 0.785]
}]
```

```
"cuboid": [{
    "name": "shape",
    "val": [12.0, 20.0, 0.0, 1.0, 1.0, 1.0, 1.0,
 4.0, 2.0, 1.5]
}]
```

```
“image": [{
    "name": "color",
    "val": "iVBORw0KGgoAA..."
    “mime_type": "image/png",
    “encoding": "base64"
}]
```

```
“mat": [{
    "name": “points3d_4xN",
    "val": [...].
    “width": 325,
    “height": 4,
    “channels": 1,
    “data_type": “uint32",
}]
```

```
“image": [{
    "name": "color",
    "val": "iVBORw0KGgoAA..."
    “mime_type": "image/png",
    “encoding": "base64"
}]
```

```
"poly2d": [
    {
        "name": "poly1",
        "val": ["5","5","1","mBIIOIII"],
        "mode": "MODE_POLY2D_SRF6DCC",
        "closed": false
    }, {
        "name": "poly2",
        "val": [5,5,10,5,11,6,11,8,9,10,5,10,3,8,3,6,4,5],
        "mode": "MODE_POLY2D_ABSOLUTE",
        "closed": false
    }
]
```

```
"poly3D" : [{
    "closed" : false,
    "coordinate_system" : "vehicle_iso8855",
    "name" : "lane_marking",
    "val" : [557.02, 29.69, -1.63, 562.51, 29.97, -
1.59, 568.00, 30.36, -1.58, 571.98, 30.76, -1.57]
}]
```

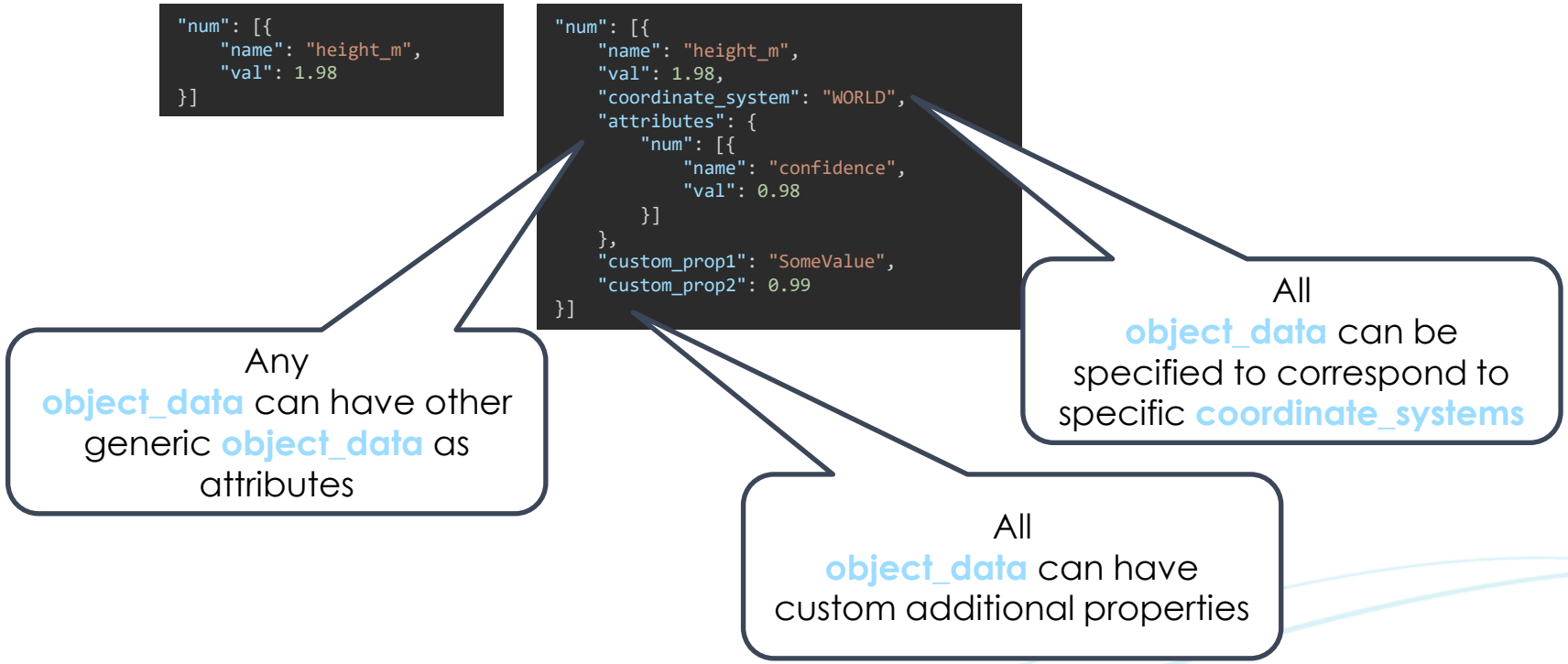
```
“point2d": [{
    "name": “A2",
    "val": [400, 200],
    “id": 1
}]
```

```
“point3d": [{
    "name": “A3",
    "val": [400, 200, 0],
    “id": 1
}]
```

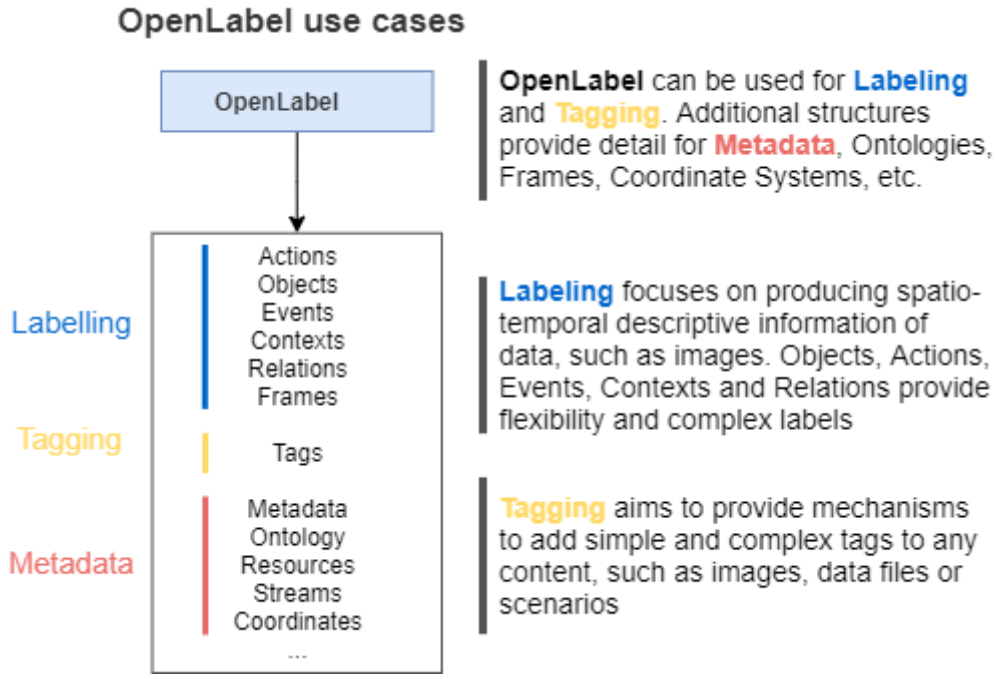
```
"mesh" : [{
    "name" : "parkslot1",
    "point3d" : {
        "0" : {
            "name" : "Vertex0",
            "val" : [25, 25, 0],
        },
        "1" : {
            "name" : "Vertex1",
            "val" : [26, 25, 0],
        },
        "2" : {
            "name" : "Vertex2",
            "val" : [26, 26, 0],
        },
        "3" : {
            "name" : "Vertex3",
            "val" : [25, 26, 0],
        },
        "4" : {
            "name" : "Vertex4",
            "val" : [27, 25, 0],
        },
        "5" : {
            "name" : "Vertex5",
            "val" : [27, 26, 0],
        }
    },
    "line_reference" : {
        "0" : {
            "name" : "Edge",
            "reference_type" : "point3d",
            "val" : [0, 1],
        },
        "1" : {
            "name" : "Edge",
            "reference_type" : "point3d",
            "val" : [1, 2],
        },
        "2" : {
            "name" : "Edge",
            "reference_type" : "point3d",
```

OpenLABEL – Data format

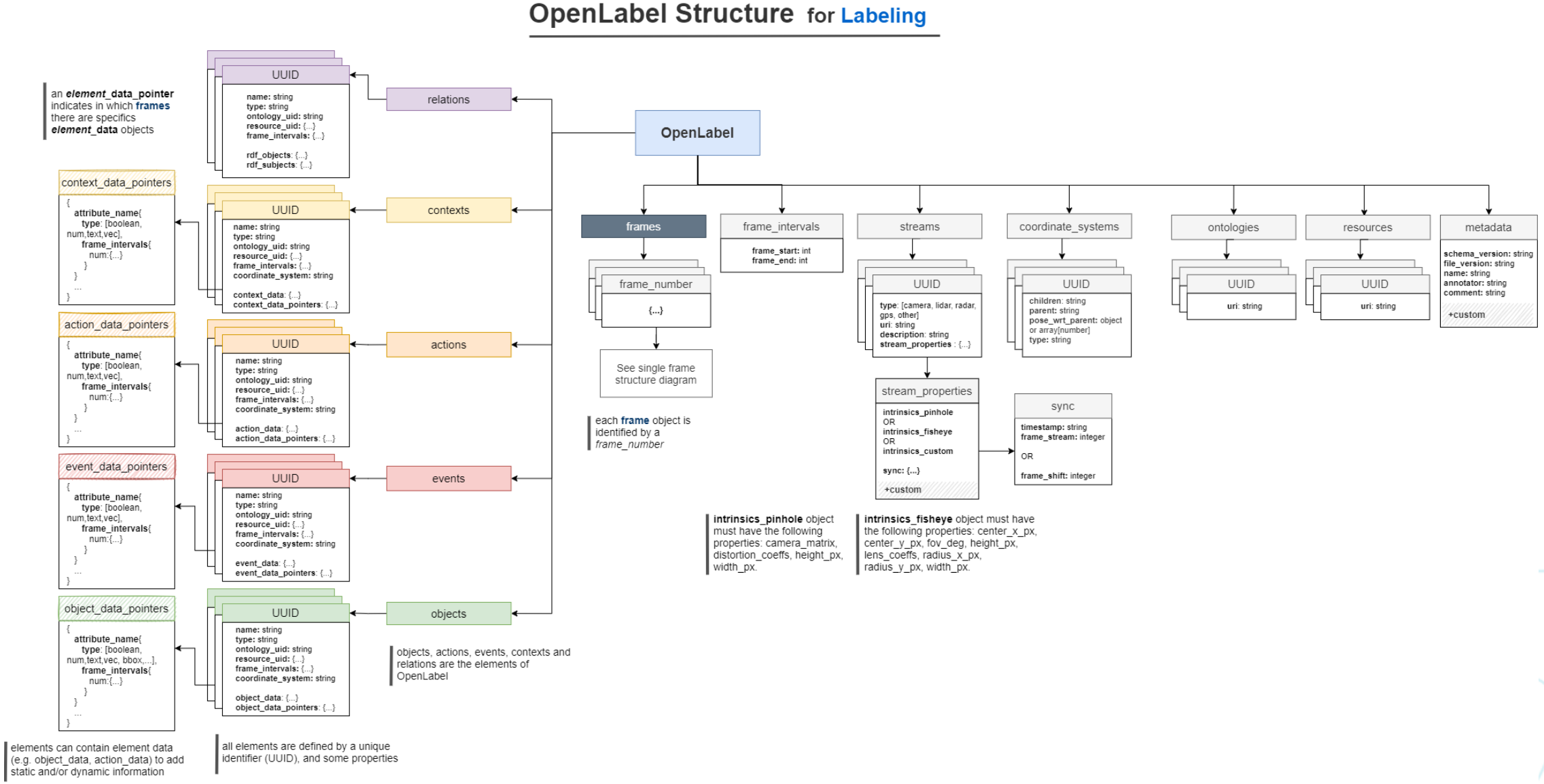
OpenLabel JSON data type extension



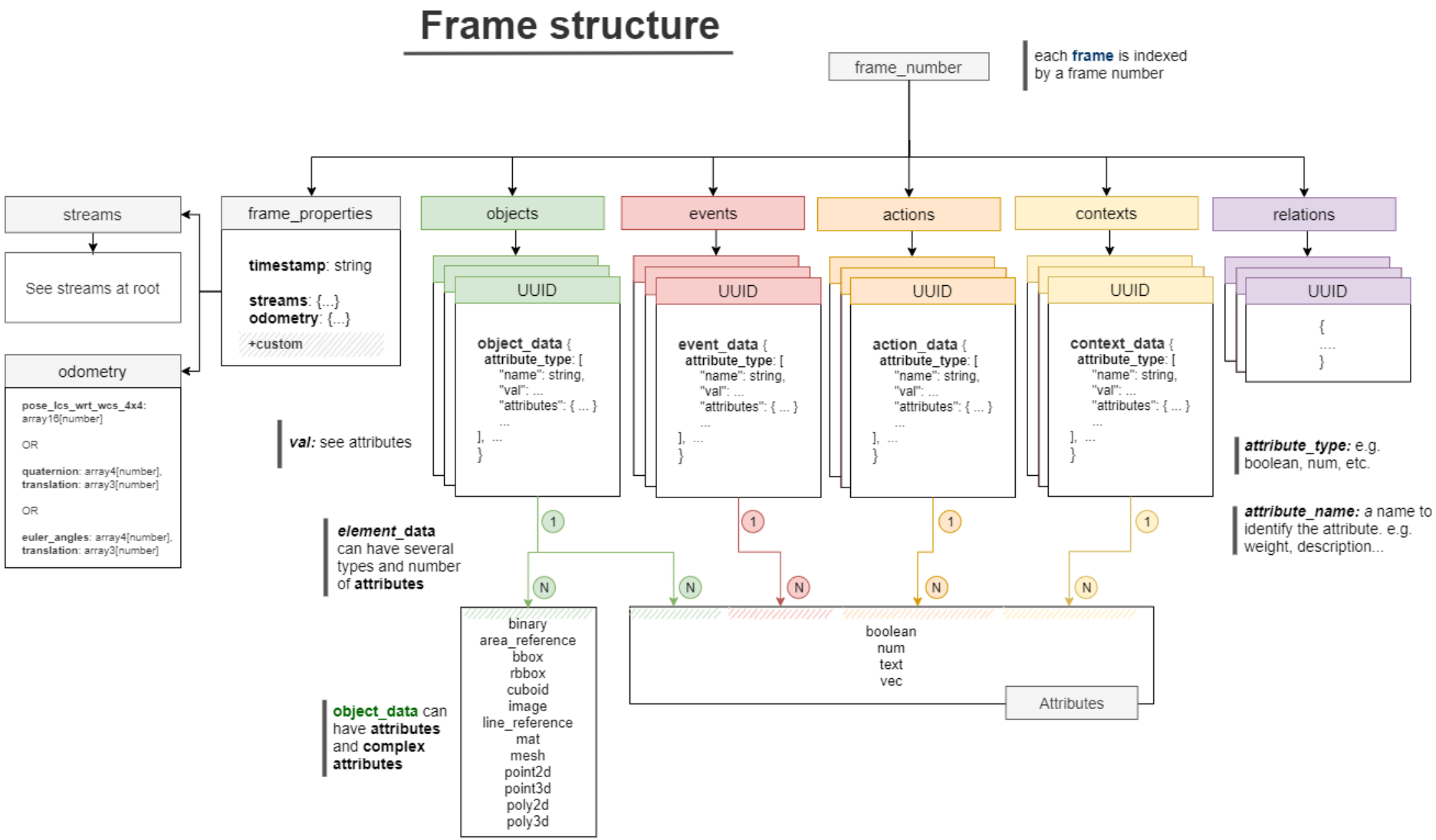
OpenLABEL – Data Format



OpenLABEL – Data Format



OpenLABEL – Data Format



OpenLABEL – Data Format

Geometric attributes

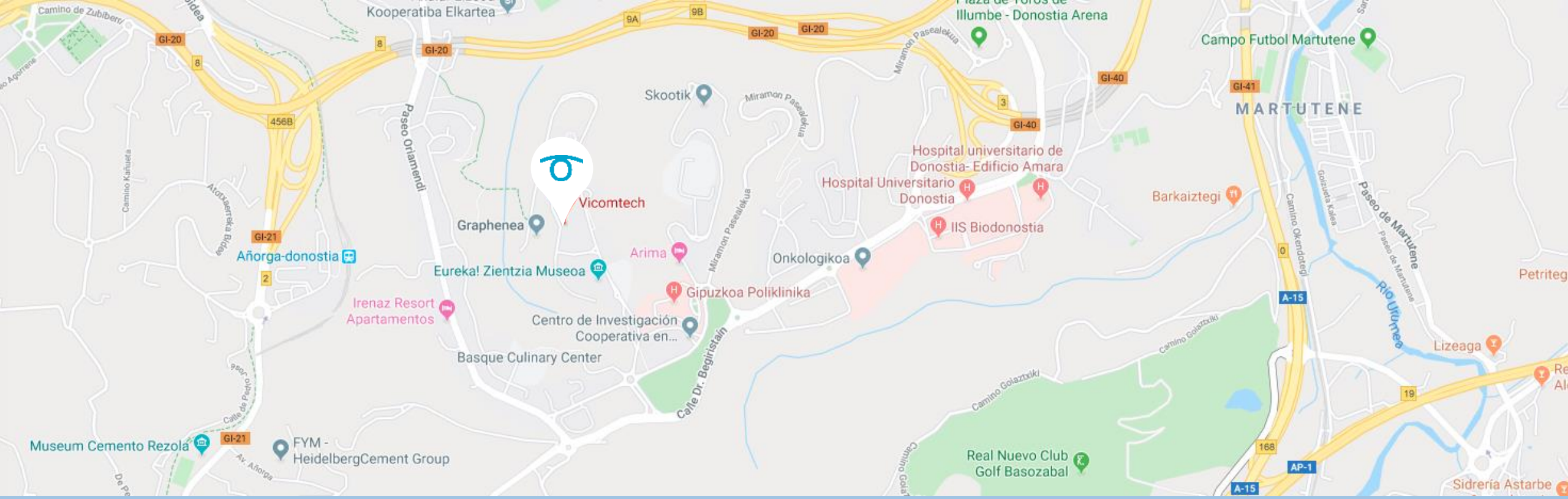
object_data			
binary	image	bbox	rbbox
coordinate_system: string **data_type:** string **encoding:** string **name:** string **val:** string **attributes:** [boolean, num, text, vec] **+custom**	**children**: string **parent:** string **pose_wrt_parent:** object or array[number] **type:** string **uid:** string	**coordinate_system:** string **name:** string **val:** string **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **name:** string **val:** array[number] **attributes:** [boolean, num, text, vec] **+custom**

object_data			
poly2d	poly3d	cuboid	mat
coordinate_system: string **mode:** string **closed:** string **name:** string **hierarchy**: array[4] **val:** array[] **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **closed:** string **name:** string **val:** array[] **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **name:** string **val:** array[9] or array[10] **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **name:** string **val:** array[number] **channels**: number **width**: number **height**: number **data_type**: string **attributes:** [boolean, num, text, vec] **+custom**

object_data		
point2d	point3d	mesh
coordinate_system: string **data_type:** string **encoding:** string **name:** string **val:** string **attributes:** [boolean, num, text, vec] **+custom**	**children**: string **parent:** string **pose_wrt_parent:** object or array[number] **type:** string **uid:** string	**coordinate_system:** string **name:** string **val:** string **attributes:** [boolean, num, text, vec] **+custom**

Attributes

element_data			
boolean	num	text	vec
coordinate_system: string **name:** string **val:** string **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **name:** string **val:** number **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **name**: string **val**: string **attributes:** [boolean, num, text, vec] **+custom**	**coordinate_system:** string **name:** string **val:** array[number OR string] **attributes:** [boolean, num, text, vec] **+custom**



Eskerrik asko
Gracias
Thank You

Address:
Paseo Mikeletegui 57
San Sebastián, Spain

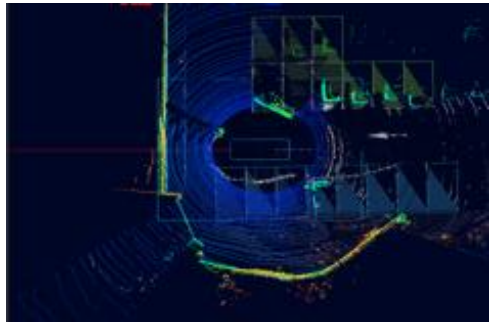
+34 943 30 92 30

mnieto@vicomtech.org

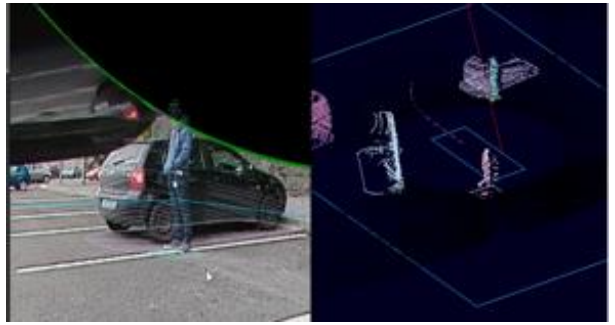
STRUCTURE

OpenLABEL WP5 – Data Format

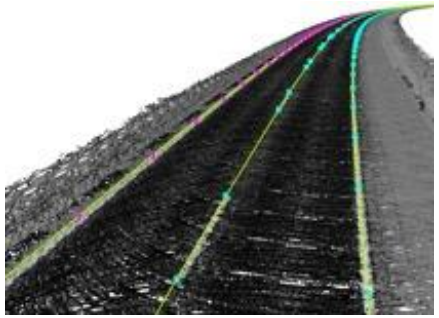
- **Annotations** which describe the content of a **scene** in an structured manner
- In many cases, **annotations** are **attached** to **data series:** videos, lidar, etc.
- **Requirements**
  - Object descriptions
  - Spatio-temporal entities
  - Synchronization and timestamps
  - Sensor calibration
  - Numerical ranges
  - Actions and events
  - Time intervals
  - Relations between elements
  - Semantic concepts



3D Parking slots



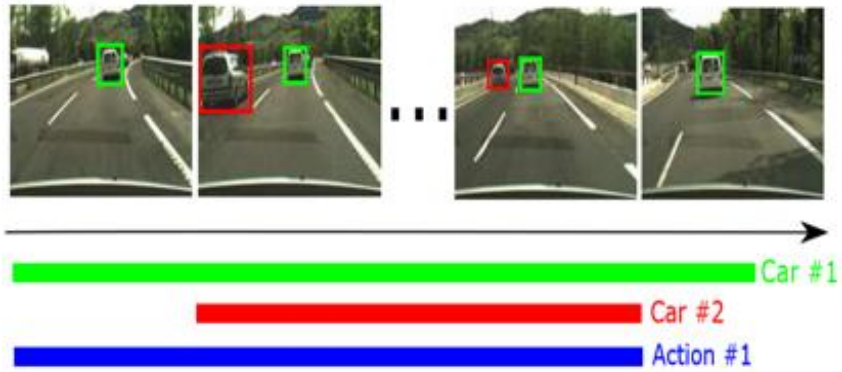
3D objects



3D lane markings

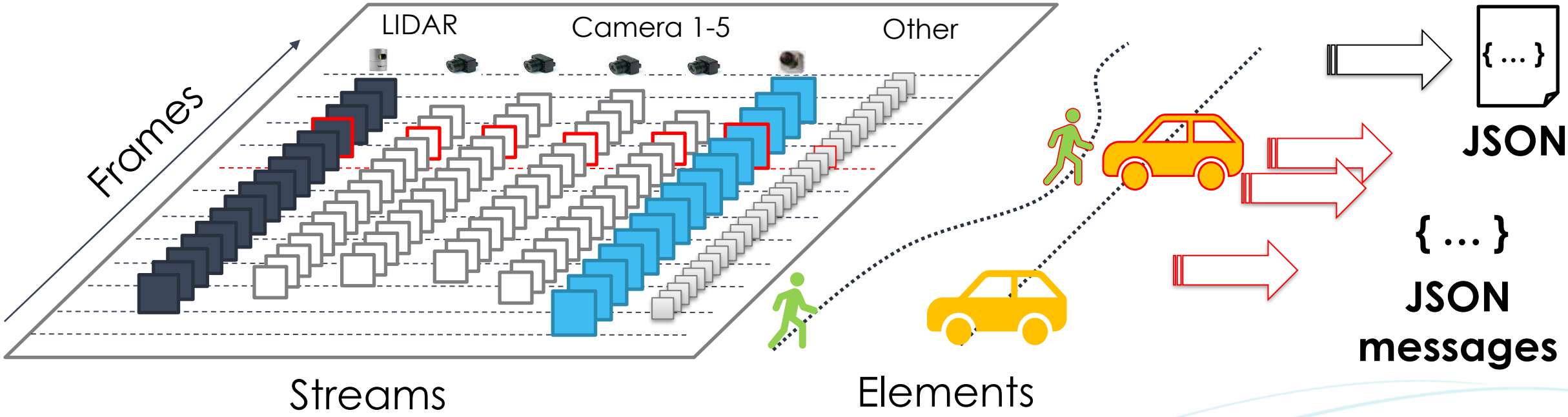
2D-3D objects

2D segmentation



Maneouvres (actions)

OpenLABEL – Data Format



OpenLABEL – Data Format

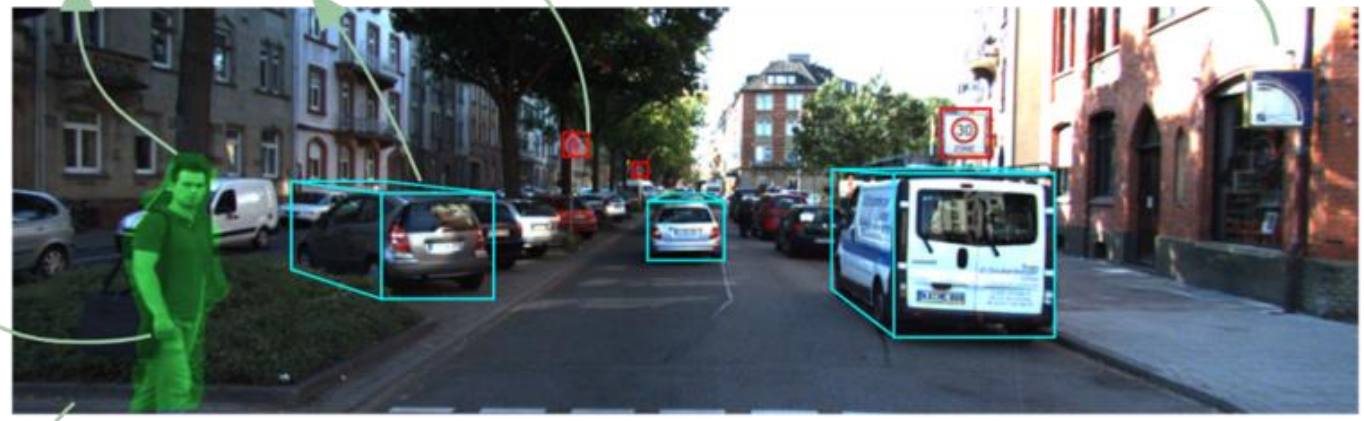
Elements = {Objects, Actions, Events, Contexts, Relations}

Object

Person, signal, car or any object with spatial description (e.g. bounding box) and sensor ID from which it is seen

Context

This is an urban scene, it is sunny it is a sequence from an onboard camera



Action

The period of time where an action happens:
looking at ego-vehicle
crossing

Event

The moment in which the person starts crossing the road

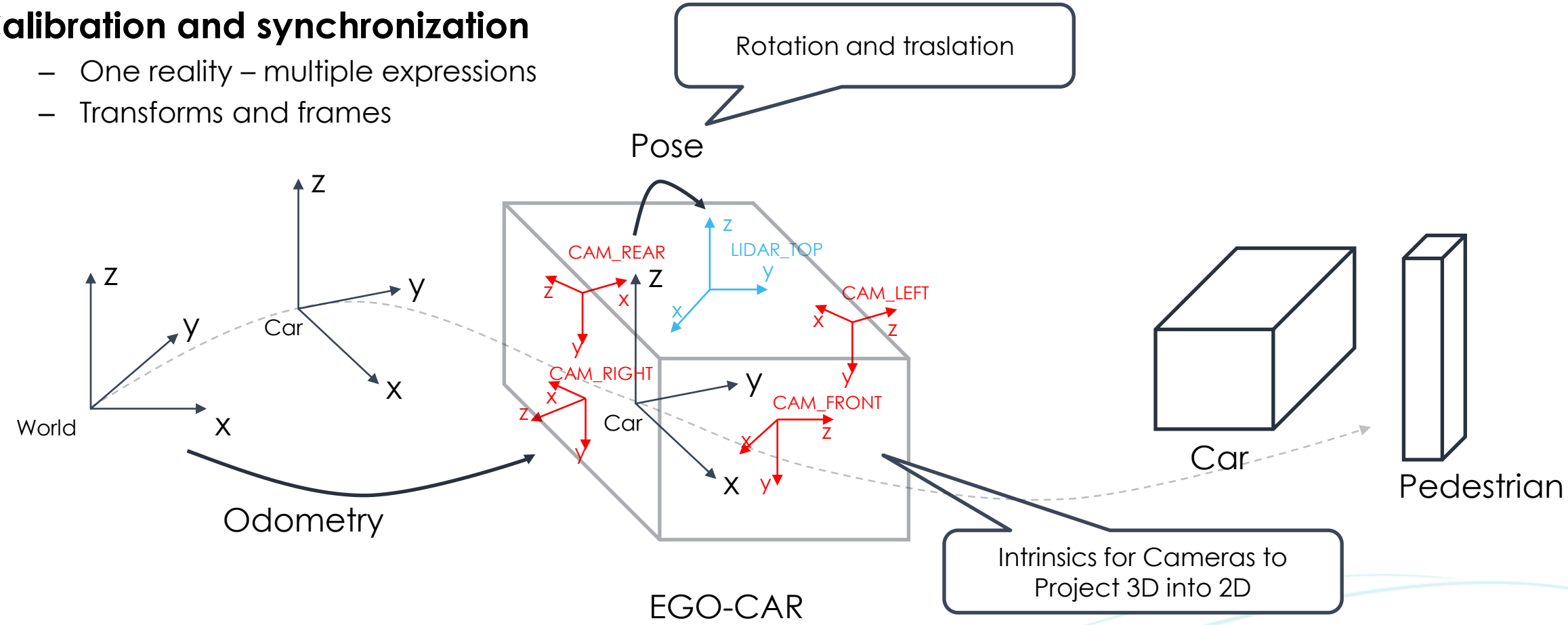
Relation

The object person is the actor of the action, and the event triggers the action.
A person crosses the road when is sunny

OpenLABEL – Data Format

Calibration and synchronization

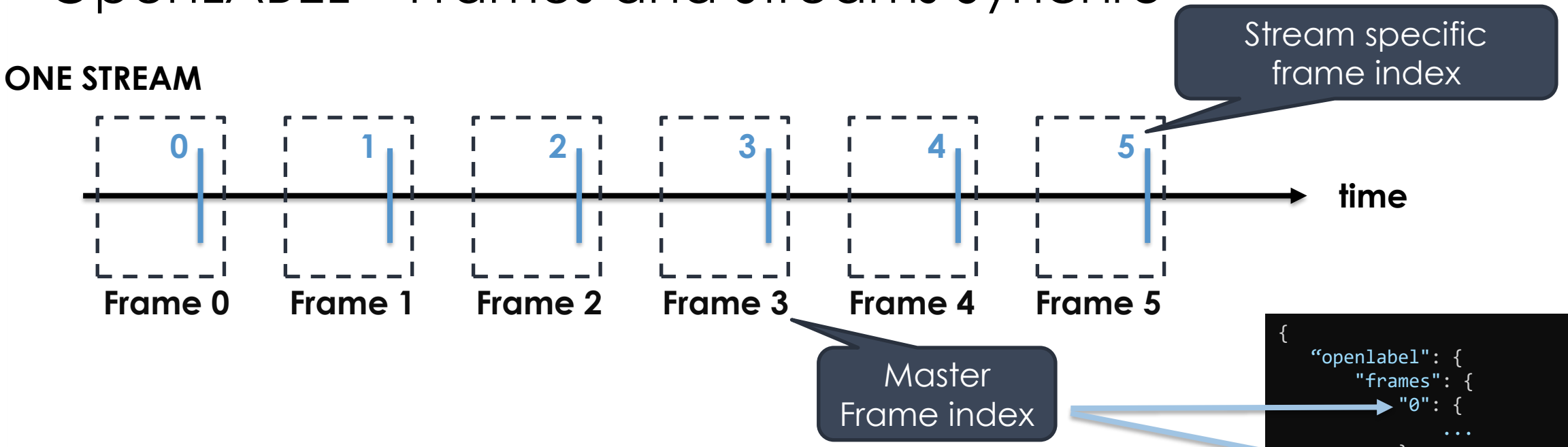
- One reality – multiple expressions
- Transforms and frames



vicomtech

SYNCHRONIZATION

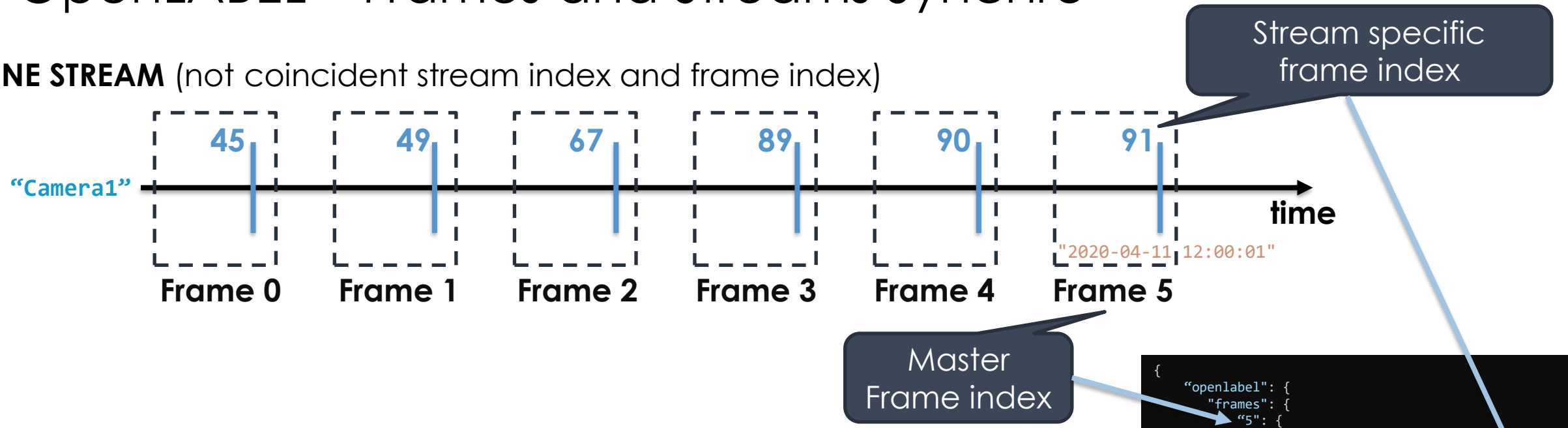
OpenLABEL – Frames and Streams Synchro



- No need to specify anything (sensor name, timestamp, etc.)
- Frame index are **integers**, starting from 0
- **Master Frame index** coincides with **Stream-specific** frame index (thus, stream-specific frame index is not labeled)

OpenLABEL – Frames and Streams Synchro

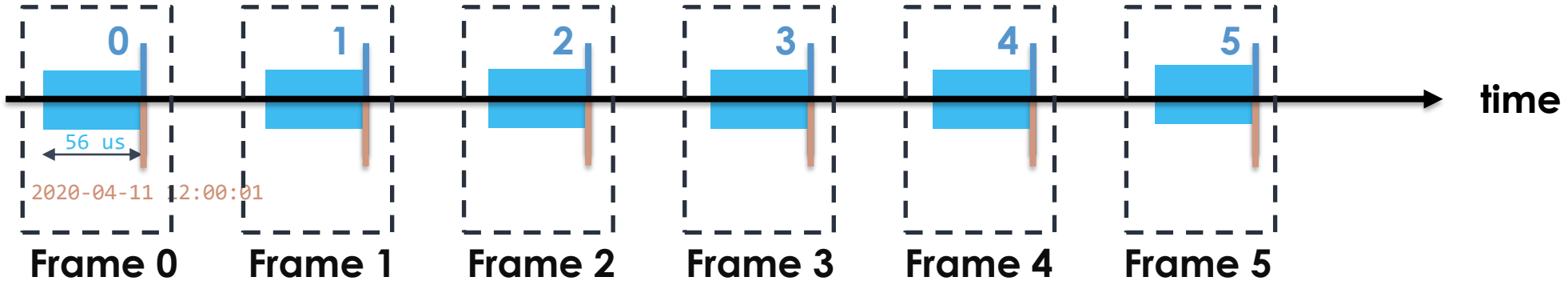
ONE STREAM (not coincident stream index and frame index)



- It is possible to have **non-coincident** Master Frame index and Stream Frame Index (which is now explicitly specified within "stream_properties" of this frame's "frame_properties"
- Other "frame_properties" can specify **timestamp** info

OpenLABEL – Frames and Streams Synchro

ONE STREAM with timestamps and other properties

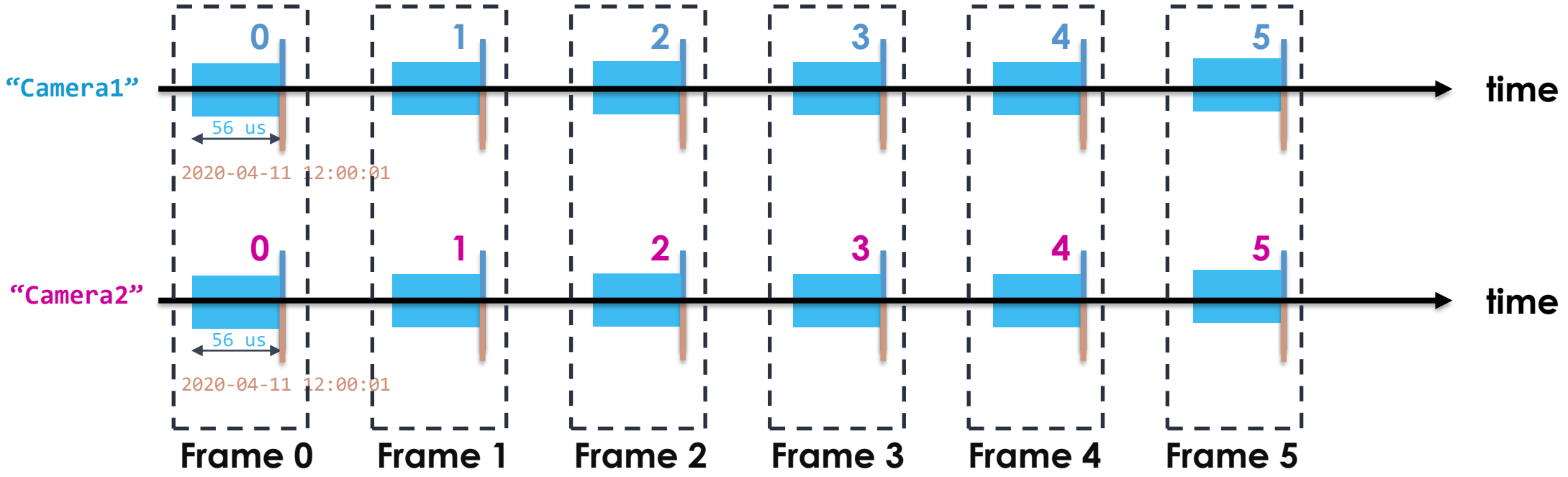


- Appart from timestamp or stream_properties, **user-specific properties** are allowed (e.g. aperture_time_us)

```
{
   “openlabel": {
       "frames": {
           "0": {
                ...
               "frame_properties": {
                   "timestamp": "2020-04-11 12:00:01",
                   “aperture_time_us": “56",
               }
           }
       }
       ...
   }
}
```

OpenLABEL – Frames and Streams Synchro

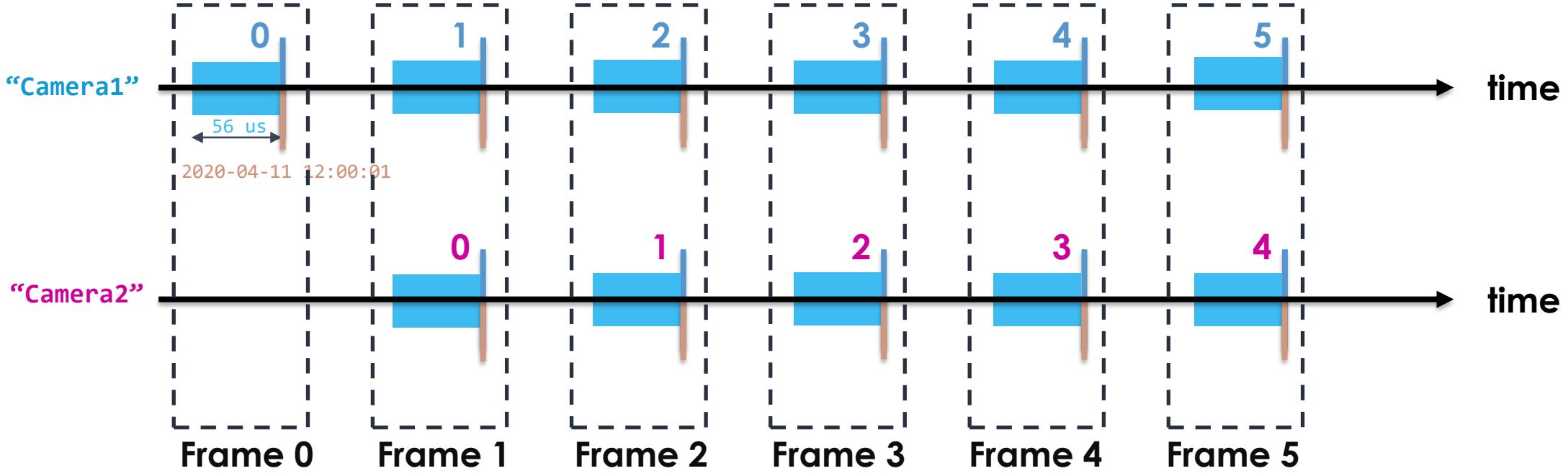
SEVERAL STREAMS (same frequency, same start and indexes)



- **Fully synchronized** case, Master Frame index coincides with each Stream Indexes

OpenLABEL – Frames and Streams Synchro

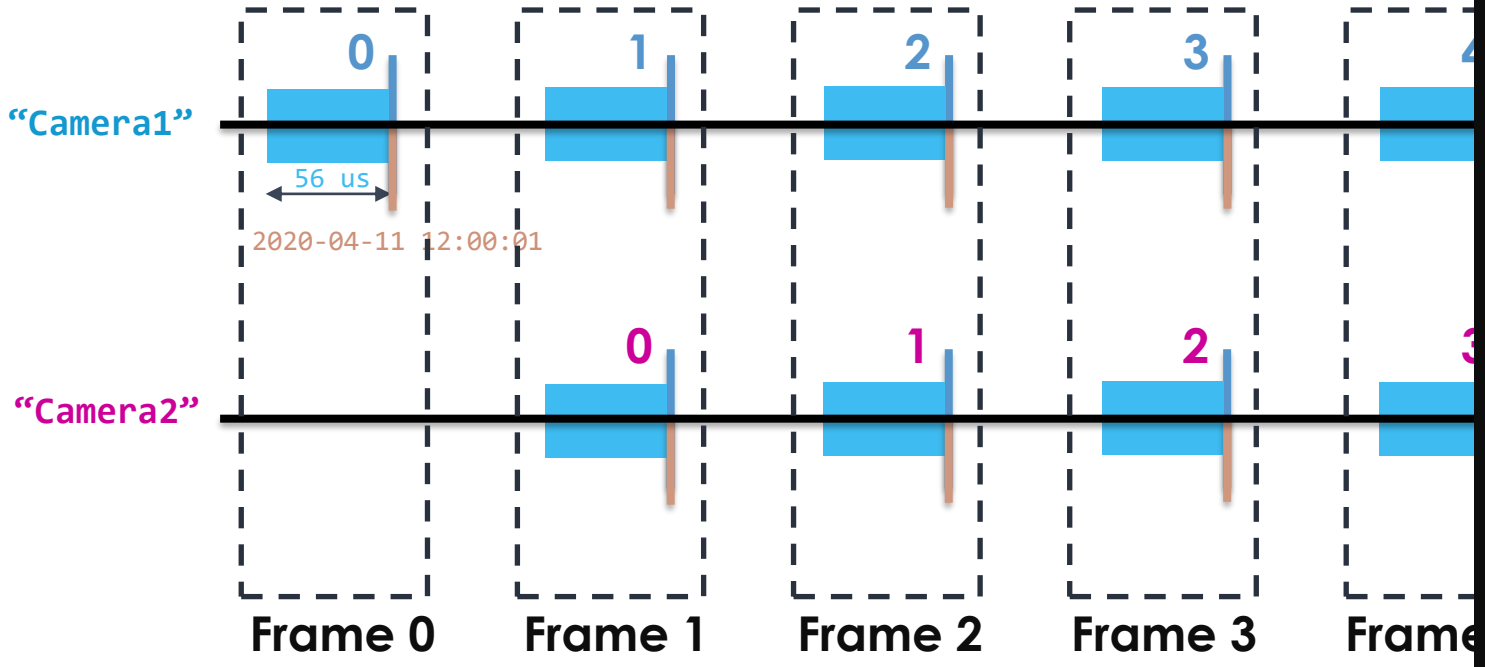
SEVERAL STREAMS (same frequency, different start or indexes)



- **Different Stream Indexes** can be enclosed into **Master Frames**
  e.g. "Camera2" starts delayed but synced

OpenLABEL – Frames and Streams Synchro

SEVERAL STREAMS (same frequency, different start or indexes)

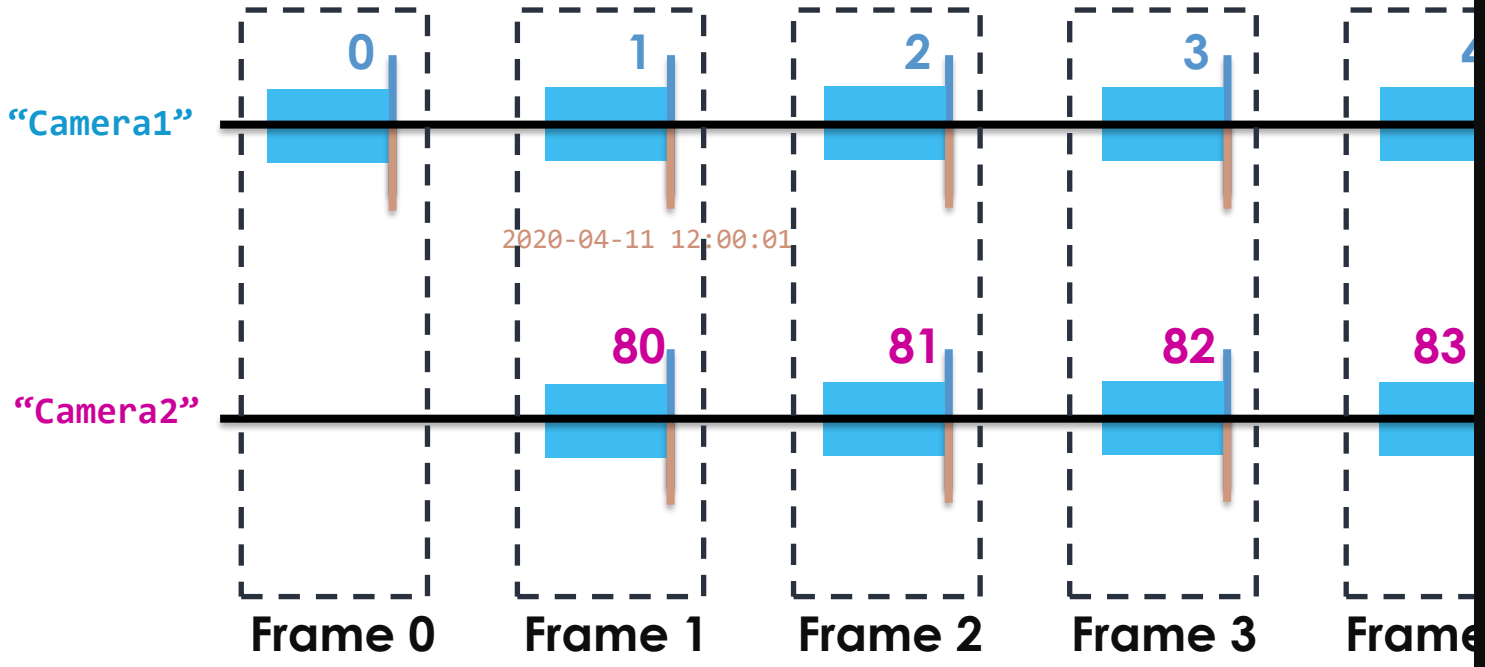



```
{
    "openlabel": {
        "frames": {
            "1": {
                ...
                    "frame_properties": {
                        "timestamp": "2020-04-11 12:00:01",
                        "streams": {
                            "Camera1": {
                                "stream_properties": {
                                    "sync": {
                                        "frame_stream": 1
                                    }
                                }
                            },
                            "Camera2": {
                                "stream_properties": {
                                    "sync": {
                                        "frame_stream": 0
                                    }
                                }
                            }
                        }
                    }
                }
            }
            ...
    }
}
```

- **Different Stream Indexes** can be enclosed into **Mas**
  e.g. "Camera2" starts delayed but synced

OpenLABEL – Frames and Streams Synchro

SEVERAL STREAMS (same frequency, different start or indexes)

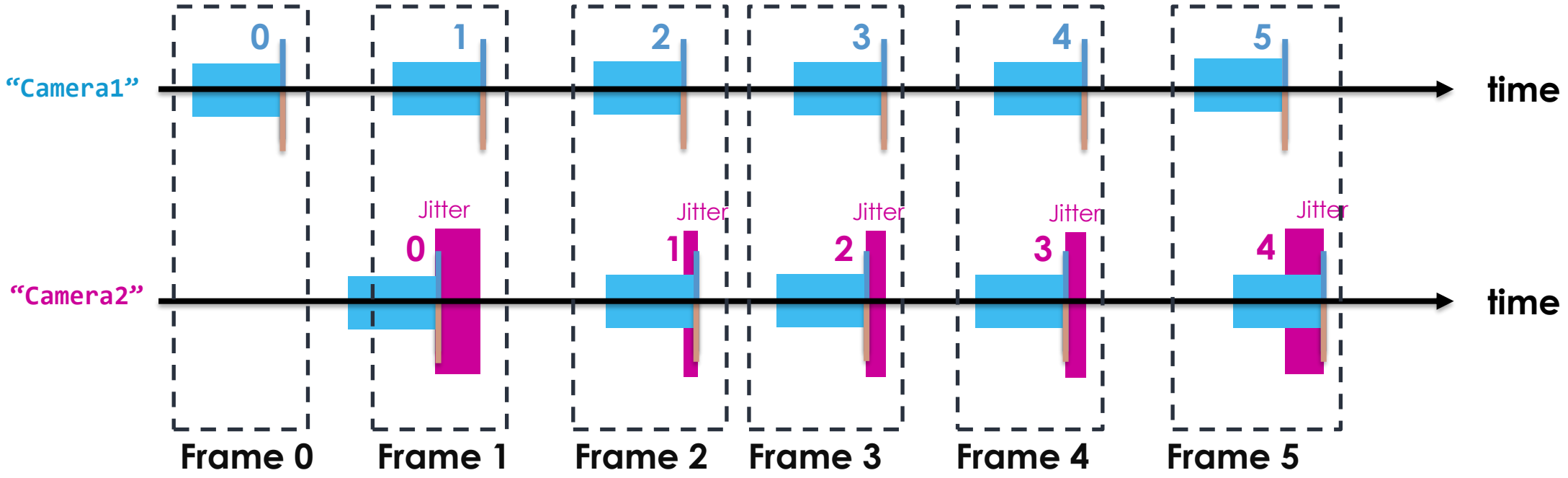



```
{
    "openlabel": {
        "frames": {
            "1": {
                ...
                "frame_properties": {
                    "timestamp": "2020-04-11 12:00:01",
                    "streams": {
                        "Camera1": {
                            "stream_properties": {
                                "sync": {
                                    "frame_stream": 1
                                }
                            }
                        },
                        "Camera2": {
                            "stream_properties": {
                                "sync": {
                                    "frame_stream": 80
                                }
                            }
                        }
                    }
                }
            }
        }
        ...
    }
}
```

- Specified inside each Master Frame to **preserve Stream Frame Index**

OpenLABEL – Frames and Streams Synchro

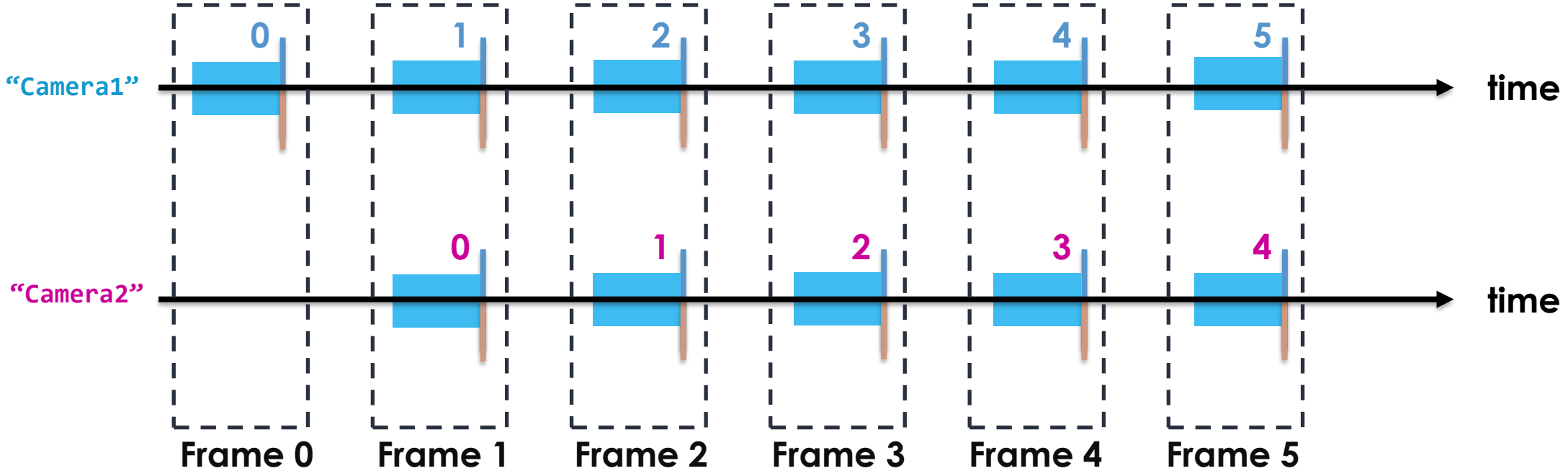
SEVERAL STREAMS (jitter)



- **Jitter**/variable frequency can also be labeled if the user provides the timestamping and the desired correspondences under the Master Frame Index

OpenLABEL – Frames and Streams Synchro

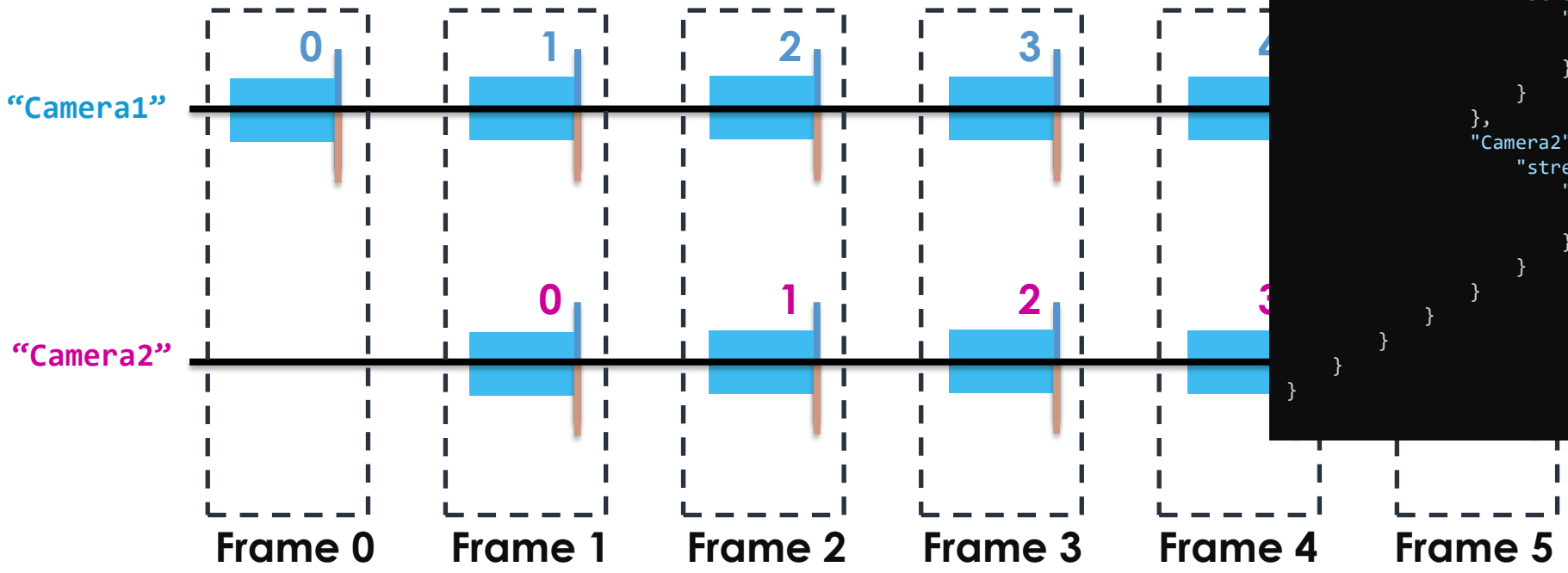
SEVERAL STREAMS (same frequency, constant shift)



- If the **shift** is known to be **constant**, a more compact representation is posible specifying the shift at root "stream_properties" and not at each frame

OpenLABEL – Frames and Streams Synchro

SEVERAL STREAMS (same frequency, constant shift)

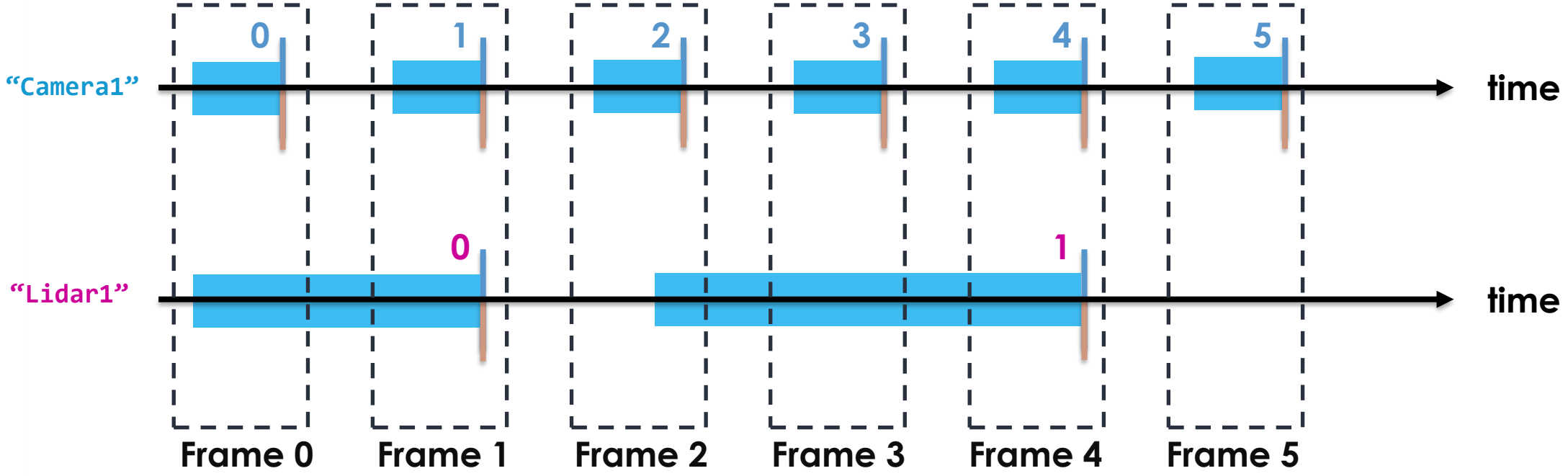



```
{
    "openlabel": {
        "streams": {
            "Camera1": {
                "stream_properties": {
                    "sync": {
                        "frame_shift": 0
                    }
                }
            },
            "Camera2": {
                "stream_properties": {
                    "sync": {
                        "frame_shift": 1
                    }
                }
            }
        }
    }
}
```

- If the **shift** is known to be **constant**, a more compact representation is posible specifying the shift at root "stream_properties" and not at each frame

OpenLABEL – Frames and Streams Synchro

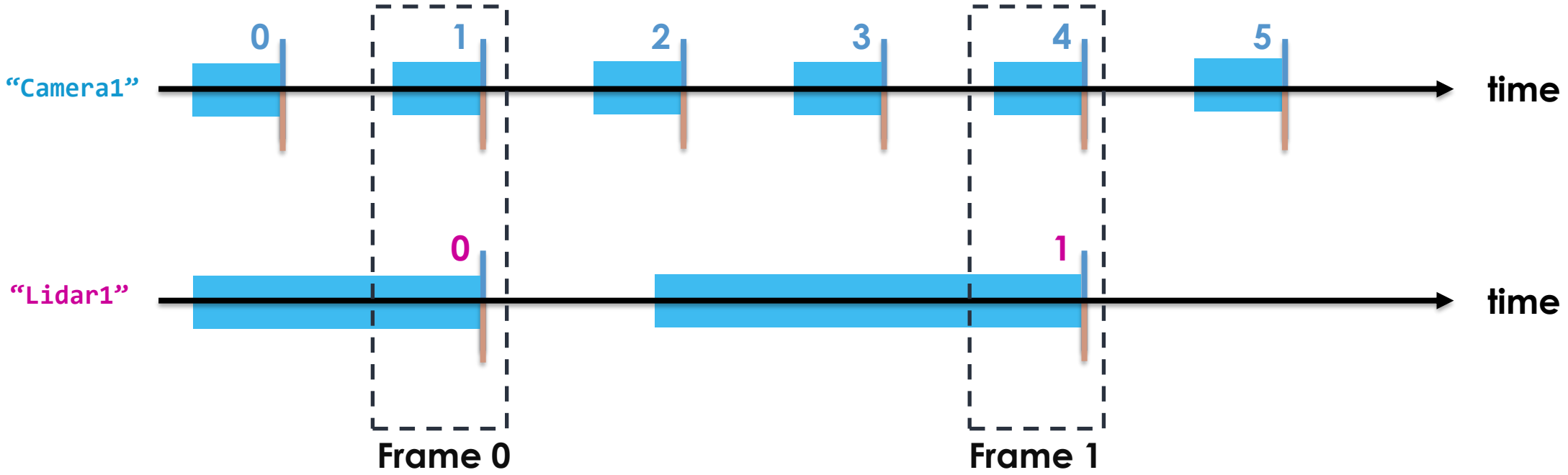
SEVERAL STREAMS (different frequency)



- Analogously to case (b), the user can specify which Stream Indexes need to be enclosed within each Master Frame
- Typically, the Master Frame Index should be defined by the highest frequency Stream

OpenLABEL – Frames and Streams Synchro

SEVERAL STREAMS (different frequency)



- The user can create a Master Frame index using the slowest Stream or any other